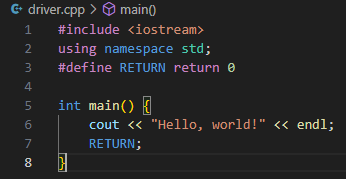
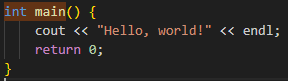
Конспект выполнил студент группы ВМК-22 Александров Данил

Этапы компиляции программы на C++

Зачем нужно компилировать исходные файлы?Исходный C++ файл — это всего лишь код, но его невозможно запустить как программу или использовать как библиотеку. Поэтому каждый исходный файл требуется скомпилировать в исполняемый файл, динамическую или статическую библиотеки

Вся работа будет происходить с файлом driver.cpp  
****

Этапы компиляции:

1. Препроцессинг  
   Препроцессор — это макро процессор, который преобразовывает вашу программу для дальнейшего компилирования. На данной стадии происходит происходит работа с препроцессорными директивами. Например, препроцессор добавляет хэдеры в код (#include), убирает комментирования, заменяет макросы (#define) их значениями, выбирает нужные куски кода в соответствии с условиями #if, #ifdef и #ifndef.  
   g++ -E driver.cpp -o driver.ii - получение препроцессированный код в выходной файл driver.ii, используя флаг -E, который сообщает компилятору, что компилировать файл не нужно, а только провести его препроцессинг  
     
   Взглянув на тело функции *main* в новом сгенерированном файле, можно заметить, что макрос RETURN был заменен
2. Компиляция  
   g++ выполняет свою главную задачу — компилирует, то есть преобразует полученный на прошлом шаге код без директив в *ассемблерный код*.  
   **Ассемблерный код** — это доступное для понимания человеком представление машинного кода.  
   g++ -S driver.ii -o driver.s - Используя флаг -S, который сообщает компилятору остановиться после стадии компиляции, получим ассемблерный код в выходном файле driver.s . Мы можем все также посмотреть и прочесть полученный результат. Но для того, чтобы машина поняла наш код, требуется преобразовать его в машинный код.
3. Ассемблирование  
   Так как x86 процессоры исполняют команды на бинарном коде, необходимо перевести ассемблерный код в машинный с помощью ассемблера.  
   Ассемблер преобразовывает ассемблерный код в машинный код, сохраняя его в *объектном файле*.  
   **Объектный файл** — это созданный ассемблером промежуточный файл, хранящий кусок машинного кода. Этот кусок машинного кода, который еще не был связан вместе с другими кусками машинного кода в конечную выполняемую программу, называется *объектным кодом*.  
   as driver.s -o driver.o – получение машинного кода с помощью ассемблера (as) в выходной объектный файл driver.o****
4. Компоновка/линковка  
   Компоновщик (линкер) связывает все объектные файлы и статические библиотеки в единый исполняемый файл, который мы и сможем запустить в дальнейшем.  
   g++ driver.o -o driver - получение исполняемого файла driver****
5. Загрузка  
   Последний этап — вызвать загрузчик для загрузки нашей программы в память. На данной стадии также возможна подгрузка *динамических библиотек*.  
   ./driver – запуск программы   
   

Итог:

1. Процессинг - получение перепроцессированного кода в файл .ii. Удаление хеддеров, комментариев и тд.
2. Компиляция – преобразование в ассемблерный код .s
3. Ассемблирование – преобразование в машинный код, сохраняя в объектном файле .o
4. Линковка – объединение всех объектных файлов и статических библиотек в исполняемый файл.exe
5. Загрузка в память/запуск исполняемого файла.

red1: процесс компиляции на примере программы simple program  
Этапы:

1. Процессинг   
   Подготовка исходного кода для дальнейших этапов компиляции.  
   main.cpp: обрабатываются директивы препроцессора, такие как #include, #define, условные компиляции (#ifdef, #endif и т.д.). Например, препроцессор откроет файл triangle.h, чтобы включить его содержимое.  
   triangle.cpp: здесь также обрабатываются директивы препроцессора. Файл triangle.h будет вставлен в начало triangle.cpp.  
   т.е будут созданы main.ii и triangle.ii
2. Компиляция  
   Преобразование в ассемблерный код   
   Создаются файлы main.s и triangle.s содержащие ассембленый код.
3. Ассемблирование   
   Создаются объектные файлы main.o и triangle.o содержащие машинный код готовый для линковки.
4. Линковка  
   Линковщик собирает все объектные файлы и библиотеки в единый исполнимый файл.  
   Линковщик соединяет объектные файлы main.o и triangle.o, разрешает вызовы функций и ссылки на переменные.  
   т.е будет создан единый исполняемый файл program.exe
5. Загрузка в память и запуск  
   Когда исполнимый файл запускается, операционная система загружает его в память:

* Операционная система выделяет область памяти для программы и загружает в нее исполнимые файлы, а также необходимые библиотеки.
* Когда программа запущена, процессор начинает выполнение кода, начиная с точки входа